REMARKS

Claims 1-3, 5-8, 10-13, 15-18, 20-23, 25-28 and 30 have been amended. No claims have been canceled or added. Accordingly, claims 1-30 remain pending in the application.

35 U.S.C. §112

The rejection of claims 7, 10, 12, 17, 20, 27 and 30 under this section has been rendered moot by the amendment of these claims.

35 U.S.C. §§102 and 103

Claims 1-4, 7-14, 17-24 and 27-30 stand rejected under 35 U.S.C. §102(b) as being anticipated by Shoup (U.S. Pub. No. 2002/0147734). Claims 5, 6, 15, 16, 25 and 26 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Shoup in view of Melahn (U.S. Patent No. 6,003,042). These rejections are traversed as follows.

Discussion of Melahn and Shoup

The Office Action cites Melahn as teaching use of a hash value to determine if a new version of a file corresponds to an older or existing version, citing column 2, lines 32-43 of Melahn. However, a closer examination of this portion of Melahn and of column 9, line 21, through column 10, line 9, shows that Melahn teaches

computing a hash table for an existing version of a file and computing a hash table for a new version of the existing file. The hash value for each record in the new version of the file is then compared with the hash value for each record in the existing file. For each record in the new version of the file having a hash value matching the hash value of a record in the existing file, the data within the corresponding record of the new version of the file is compared with the data within the corresponding record of the existing file. "Copy" scripts containing various length and location information about the records in the new version of the existing file that match records in the existing file are created. In addition, "add" scripts containing the records in the old version of the existing file that do not match records in the new version and length information about the non-matching records are also created. The old version is then discarded, having been replaced by scripts which contain all the information necessary to regenerate the old version from the new version. The new version and the scripts are stored (see, e.g., column 2, lines 30-54).

Thus, under the method taught by Melahn, the hash values are calculated for reducing the amount of stored data by determining redundancy between an existing version of a file and new version of the file. This procedure is illustrated in FIG. 6, and following completion of the procedure, the old version of the file is discarded, the new version of the file and any scripts are stored, and there is no longer a need for the computed hash values.

The present invention has both a different purpose and operation from that taught by Melahn. In the present invention, a first hash value is calculated for the

original file and stored with an original inode for the original file. The original inode also stores information regarding one or more secondary inodes for format converted files. Each format-converted file is identified by one of these secondary inodes. An additional hash value is calculated for each format-converted file and stored with the secondary inode for that format-converted file (see, e.g., pages 16-17, Tables 6A-6B and accompanying text of the specification of the present application). However, unlike Melahn, the first hash value and the additional hash values are not compared with each other, since this would be meaningless because the hash values are for files of different file formats that do not contain similar data on a bit level.

Rather, the purpose of the invention is for long term data archiving. Thus, determining the status of the stored files and the format converted files is performed by reading the stored files, calculating new hash values for the stored files, and comparing the new hash values with the hash values previously stored with the inodes. Under the invention, as discussed at pages 23-24 of the specification, a requestor can obtain the status of the files and the format-converted files. The storage system reads an inode of the specified directory, and for each file listed in the directory, the storage system reads an original inode of a file. The storage system reads the file pointed to by the original inode and calculates a new hash value for the read file. The storage system compares the newly calculated hash value with the hash value stored in the original inode to determine if the file is unchanged, uncorrupted, etc. The storage system then reads each secondary inode listed in the original inode for format-converted files. The system then reads each

format-converted file pointed to by a secondary inode, calculates a new hash value for the read format-converted file, and compares the new hash value with the hash value stored in the secondary inodes to determine whether the format converted files are unchanged. In this manner, the storage system may, whenever requested, determine the status of the original and format-converted files, create a list, and assign statuses, such as is shown in table 4 on page 11 of the specification.

Accordingly, the present invention sets forth a system and method for archiving both original files and format converted files by using inodes and hash values for managing the relationship and status of the archived files. By using the hash values stored with the inodes, a user may obtain a current status of the archived files. Further should one of the original files be changed, corrupted, or the like, the user will be able to determine if a format converted file is unchanged and able to be reconverted to the original file. Shoup only teaches the storing of files in multiple formats, and does not teach the present invention's method of managing the files or detecting changes in any of the files. Thus, neither Melahn nor Shoup teach the present invention's method of archiving files in multiple formats, managing the relationships between the archived files, and providing the ability to check and update the statuses of the archived files or restore archived files if a status is shown to be changed.

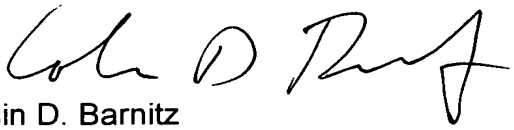
The claims of the present application have been amended to more set forth these aspects of the present invention. Neither Melahn, nor Shoup, nor any combination thereof teaches or suggests the features of the invention set forth in the

amended claims. Thus, Melahn and Shoup do not teach or suggest that a first hash value is used to determine whether the original file has changed and/or a second hash value is used to determine whether the format converted file has changed, as set forth in amended claims 1 and 11. Additionally, Melahn and Shoup do not teach or suggest managing a relationship between an original file and a format converted file by storing in a first inode a pointer to the original file and an inode number of a second inode, wherein the second inode points to the format converted file, as set forth in amended claim 21. Accordingly, independent claims 1, 11 and 21 are allowable. The remaining claims depend from these claims, claim additional patentable features of the invention, and are also allowable at least because they depend from allowable base claims.

CONCLUSION

In view of the foregoing, Applicant respectfully requests that a timely Notice of Allowance be issued in this case.

Respectfully submitted,



Colin D. Barnitz
Registration No. 35,061

MATTINGLY, STANGER, MALUR & BRUNDIDGE, P.C.
1800 Diagonal Rd., Suite 370
Alexandria, Virginia 22314
(703) 684-1120
Date: February 28, 2006